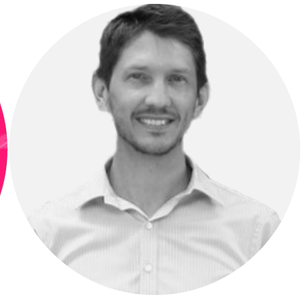
# Testing Component Rendering

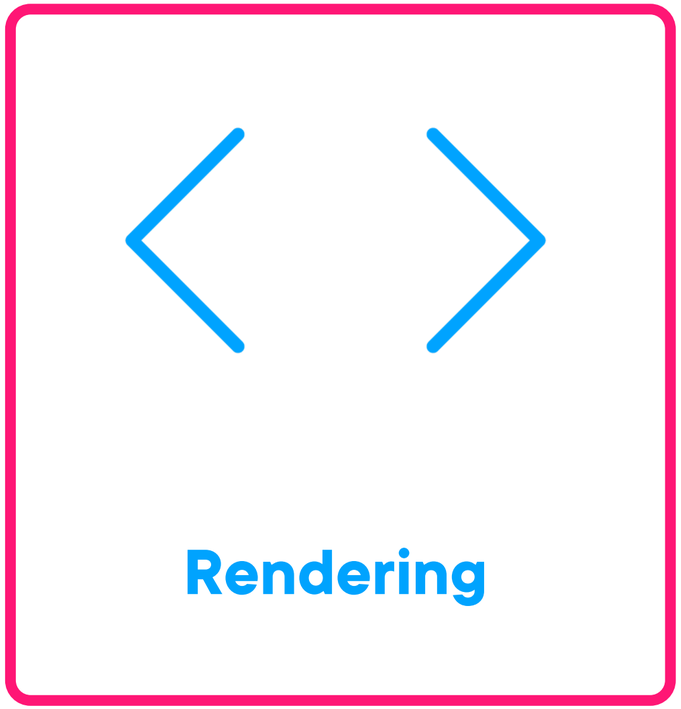



**Liam McLennan**

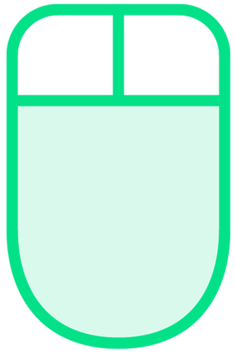
Principal Software Engineer

@liammclennan | withouttheloop.com

# What Are We Testing?



Rendering



User Interaction

# Required Tools

JavaScript runtime

JavaScript test runner

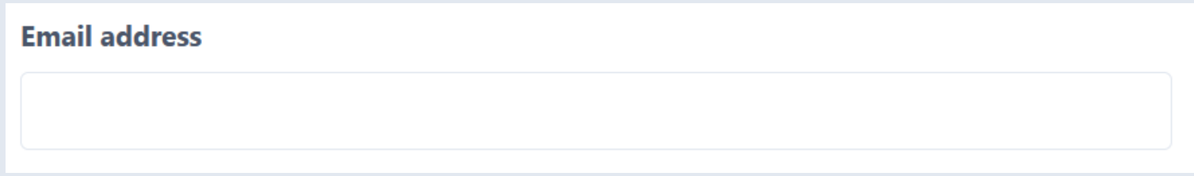
Test assertion library

DOM and HTML environment

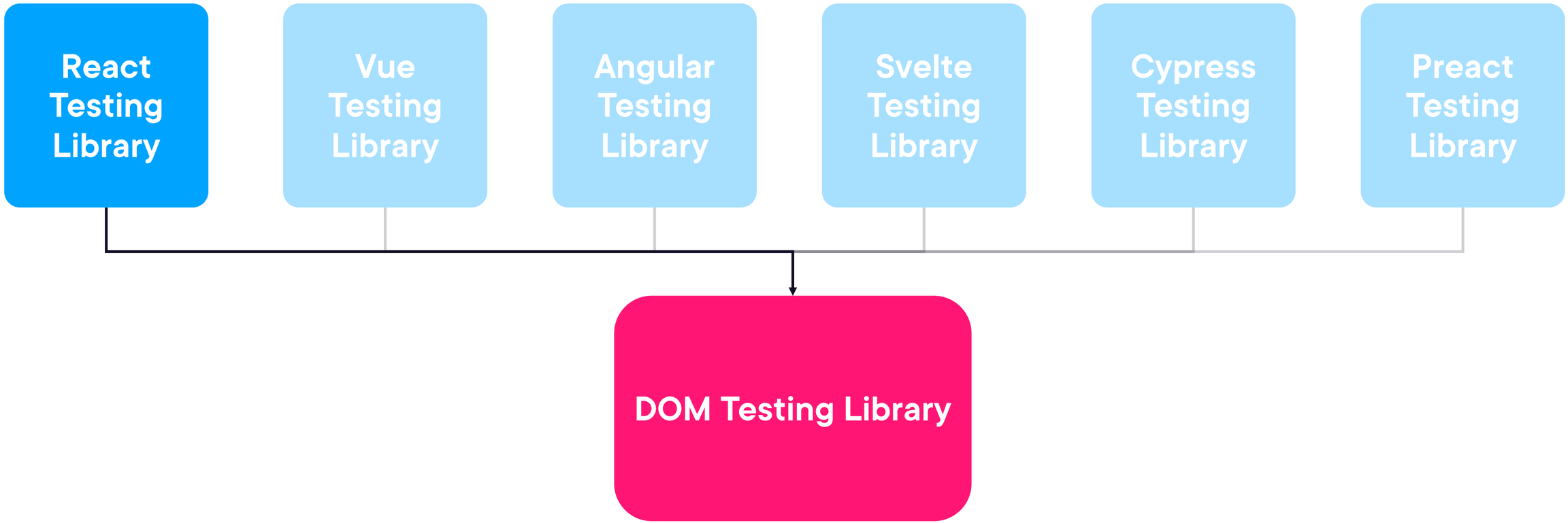
UI testing library

# User-centric Testing

```
<label for="email">
    Email address
</label>
<input
    type="email"
    id="email"
/>
```



# Testing Library



# Rendering a Component

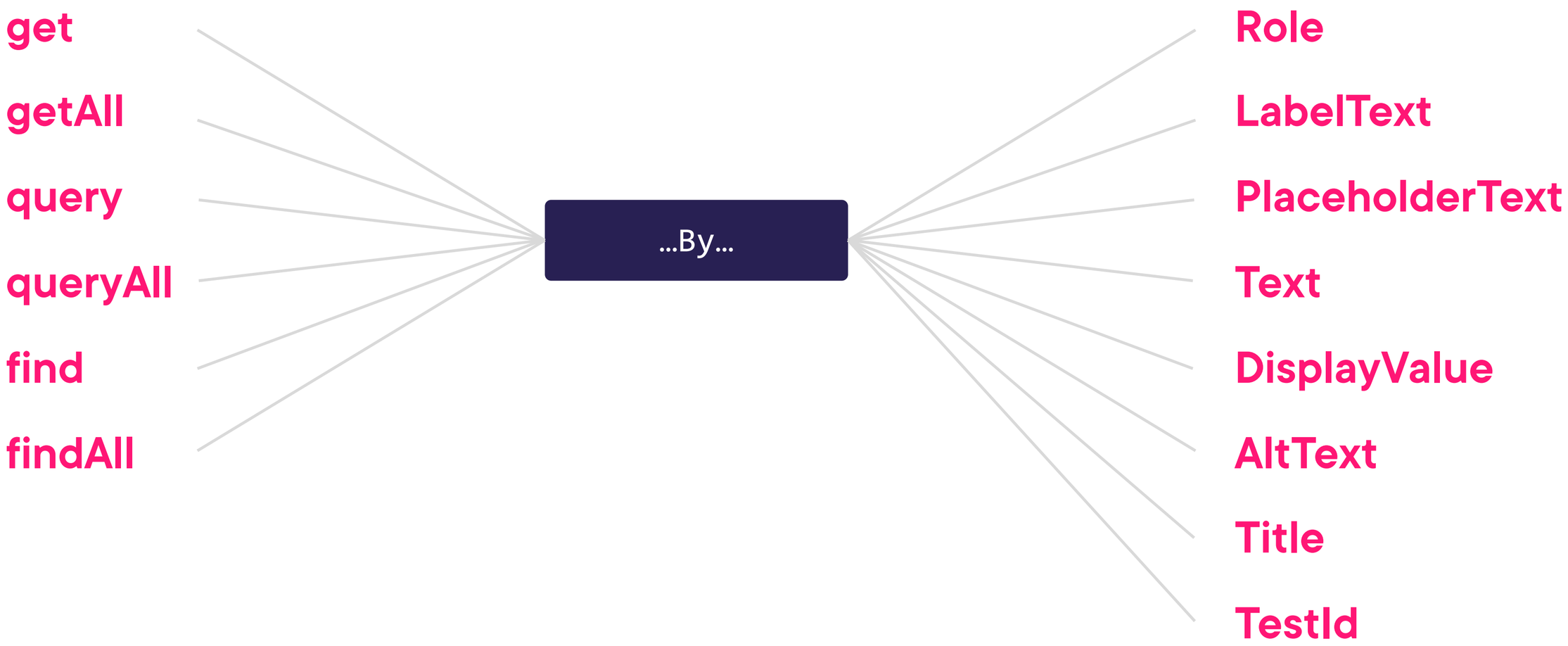
```
import { render } from '@testing-library/react'

render(<Home />)
```

# Element Query Types

| | |
|---|---|
| get | Throws an error unless there is exactly *one* match |
| getAll | Returns all matches. Throws an error if there are no matches. |
| query | Return matching node. Throws an error if there are multiple matches. |
| queryAll | Return all matches. |
| find | Asynchronous version of `get`. |
| findAll | Asynchronous version of `getAll`. |

# Element Selection Types



# ARIA Role Types

```
const heading = screen.getByRole('heading', {
  name: "Filter List",
})
```

# Test That an Element Renders

```
import { render, screen } from '@testing-library/react'
import Home from '@/pages/index'

test('Home renders a heading', () => {
  render(<Home />)

  screen.getByRole('heading', {
    name: "Filter List",
  })
});
```

# Finding Nested Elements

```
import { render, screen, getByRole } from '@testing-library/react'
import Home from '@/pages/index'

test('Home renders a heading nested',() => {
  render(<Home/>);

  const main = screen.getByRole('main');

  getByRole(main, 'heading', {
    name: "Filter List"
  });
});
```

# TextMatch

Input.js

```
<label htmlFor="name">Name</label>
<input type="text" id="name" />
```

Input.test.js

```
screen.getByLabelText("name");
screen.getByLabelText(/^name$/i);
```

# Demo

## Testing Rendering

# Debugging Components

```
screen.debug(screen.getByPlaceholderText("Filter"));
```

```
<input
  name="Filter"
  placeholder="Filter"
  type="text"
  value=""
/>
```

# Logging ARIA Roles

```
const list = screen.getByRole('list’, {
  name: /Filtered list/I
});
logRoles(list);
```

```
list:

    Name "Filtered list":
    <ul
      aria-label="Filtered list"
    />

------------------------------
listitem:

    Name "":
    <li />

    Name "":
    <li />
```

# Log Playground URL

```
render(<FilterList {...props} />);
screen.logTestingPlaygroundURL();
```

https://testing-playground.com/#markup=DwEwlgbgfKkAQGMA2BDAzmgvAIgGZiQBcBTAJyTDUOxgAcYwA7WgV0LkZQFticAxAiVLY4hAJ61e2EgA9qcWqgTEAFgHskIMv0FkREFEhZSawAPT1gLJHBSkwKALSoARsSQ6iZYiDgUqphRQALJkCCykYuZBwEEAasSMLGjRYDBBAKJ2hCqp6WnBdilmMUEAUiy0YEJ5sWkAyiiEEYy1QQCqpChJxaVpAHLEtM2MxLVm1jBm4NDmM1BAA

# Testing Asynchronous Rendering

App.js

```
<LazyLabel delay={500}>
  Some lazy-loaded content
</LazyLabel>
```

App.test.js

```
render(
  <LazyContent delay={500}>
    ... or not to be
  </LazyContent>
);

await screen.findByText('... or not to be');
```

# Testing Asynchronous Rendering

App.js

```
<LazyLabel delay={500}>
  Some lazy-loaded content
</LazyLabel>
```

App.test.js

```
render(
  <LazyContent delay={500}>
    ... or not to be
  </LazyContent>
);

await waitFor(() => {
  screen.getByText('... or not to be’);
});
```

# Summary

Test UI components in a user-centric way